Introduction to Optimization

Valentina Cacchiani

Department of Electrical, Electronic and Information Engineering
“Guglielmo Marconi”,
University of Bologna,
Italy

July 2023

Table of contents

- Robust optimization
- Seminar: models and heuristic algorithms for real-life applications

Robust optimization

Data Uncertainty

- Uncertainty of data is common in practice
- A solution that is optimal for a given input may even be infeasible for different data
- As a consequence, the planned solution has to be changed in real-time, leading to costs and inconvenience
- On the contrary, if in planning we can account for "some" uncertainties, then the solution may still be optimal or at least feasible during operations
- Robustness deals with the decision of accounting for uncertainties in planning so that the plan reamins feasible during operations

Efficiency and Robustness

- To avoid replanning during operations, an option could be to make a conservative choice that takes into account all bad situations that may occur
- In this way, operations will run smoothly
- However, this is in contrast with providing an efficient solution: goals of the nominal problem
- Hence, a trade-off between the nominal and the robust problems must be achieved:

An example: Train Timetable Robustness

- The aim of robustness is to determine timetables that perform well under disturbances
- A common way to obtain robust timetables is to introduce in the planning phase buffer times that can absorb possible delays occurring at an operational level
- Buffer times correspond to empty time slots used to mitigate delay propagation
- Robust Train Timetabling calls for determining where the buffer times should be inserted and how long they should be to guarantee a good trade-off between the nominal efficiency and the delay resistance

Two Classical Robustness Paradigms

Soyster (1973)

- The seminal work by Soyster (1973): strict robustness
- later extended in Ben-Tal and Nemirovski (1998)
- deals with the uncertain data present in a mathematical model
- Determine a solution that is feasible for all the considered scenarios, with the goal of minimizing the worst-case performance of a solution
- These methods tend to be overconservative

Bertsimas and Sim (2003,2004)

- Bertsimas and Sim (2003,2004): uncertainties of data are represented by letting each coefficient assume a value in an interval centered in its nominal value
- The number of coefficients that can simultaneously take their worst-case value is limited
- Define a robust model such that its optimal solution is feasible for every change of at most Γ_i coefficients in each row i of the constraint matrix
- These robust solutions can be considerably worse w.r.t efficiency than the nominal ones, even if few coefficients are allowed to change in each row

Bertsimas and Sim (2003,2004)

$$\min\{\sum_{j\in N} c_j x_j : \quad \sum_{j\in N} a_{ij}x_j \leq b_i \ \ i \in M, \quad x_j \geq 0, \ j \in N\}$$

Robust counterpart:

$$\sum_{j\in N} a_{ij}x_j + \beta(x,\Gamma_i) \leq b_i \quad i \in M$$

where

$$\beta(x,\Gamma_i) = \max_{S\subseteq N:|S|\leq\Gamma_i} \sum_{j\in S} \hat{a}_{ij}x_j$$

Recent Robustness Paradigms

- Recoverable Robustness
- Light Robustness

Recoverable Robustness

Recoverable Robustness

- Liebchen, Lübbecke, Möhring, and Stiller (2009)
- Integrates the notion of robustness and recoverability (delay management)
- An optimization problem which in a limited set of scenarios can be made feasible, or recovered, by a limited effort
- Recovery actions can be used to make a plan feasible through limited changes in every likely scenario
- Typical recovery actions: delaying events, cancelling connections, cancelling train services or rerouting trains

Recoverable Robustness

- Given on input: a set of likely scenarios, the nominal plan, and a set of recovery algorithms
- A solution is recovery-robust if, in all the considered scenarios, one can recover the solution by means of one of the given recovery algorithms
- For practical purposes, one must impose sensible limits on the recovery algorithms
- Two-stage approach

An example: Recoverable Robustness for Train Timetabling

- A set S of delay scenarios (small disturbances)
- Nominal plan: v_i $(i \in E)$ event times of the planned timetable
- Recovery actions: delaying events
- $\tilde{v}_{is}$ $(i \in E,\ s \in S)$: event times of the realized timetable in scenario s
- Constraints to respect the minimum process times when scenario $s \in S$ occurs:

$$\tilde{v}_{js} - \tilde{v}_{is} \geq l_{ijs} \quad ((i,j) \in A, s \in S)$$

An example: Recoverable Robustness for Train Timetabling

- Constraints to model the recovery action of event delaying:
  - An event in scenario $s \in S$ cannot take place before its planned time:
  $$\tilde{v}_{is} \geq v_i \quad (i \in E_d, s \in S)$$
  - limit the weighted sum of the delays of all the arrival events (w_i number of passengers of event i)
  $$\sum_{i \in E_a} w_i(\tilde{v}_{is} - v_i) \leq \lambda_1$$
  - limit the delay for each arrival event separately
  $$\tilde{v}_{is} - v_i \leq \lambda_2, \quad i \in E_a.$$
- λ_1 and λ_2: minimized in the objective function

An example: Recoverable Robustness for Train Timetabling

- Advantages: combines robustness and recoverability, thus overcoming the drawbacks of strict robustness
- Disadvantages: the recovery algorithms have to be included in the optimization model, hence only limited recovery actions can be taken into account

Light Robustness

Light Robustness

- Fischetti and Monaci (2009)
- Single stage approach (without scenarios)
- A protection level (e.g., buffer time) is required for each activity but slack variables are introduced to compensate for the missing protection
- Goal: to minimize the sum of the slack variables, while imposing a limit on the worsening of the nominal efficiency

Light Robustness

- γ_i: slack variable
- $\beta(x, \Gamma_i)$: to decide the protection level
- δ: to decide the maximum worsening compared to the nominal efficiency
- z^*: nominal solution value

Light Robustness

$$\min \sum_{i \in M} w_i \gamma_i \tag{1}$$

$$\sum_{j \in N} a_{ij} x_j + \beta(x, \Gamma_i) - \gamma_i \leq b_i \quad i \in M \tag{2}$$

$$\sum_{j \in N} a_{ij} x_j \leq b_i \quad i \in M \tag{3}$$

$$\sum_{j \in N} c_j x_j \leq (1 + \delta) z^* \tag{4}$$

$$x_j \geq 0 \quad j \in N \tag{5}$$

$$\gamma_i \geq 0 \quad i \in M \tag{6}$$

An example: Light Robustness for Train Timetabling

- v_i (with $i \in V$): time instant of event i
- l_{ij} (with $(i,j) \in A$): minimum time difference between the two consecutive events i and j
- γ_{ij}: slack variables
- Δ_{ij}: required protection level parameters
- δ: maximum worsening of the objective w.r.t the nominal efficiency F^*

$$\min \sum_{(i,j)\in A} \gamma_{ij}$$

$$v_j - v_i \geq l_{ij} \ \forall (i,j) \in A$$

$$v_j - v_i + \gamma_{ij} \geq l_{ij} + \Delta_{ij}, \quad \forall (i,j) \in A$$

$$F(v) \leq (1+\delta)F^*$$

$$\gamma_{ij} \geq 0 \ \forall (i,j) \in A$$

Light Robustness

- Advantages: faster than stochastic programming approaches and accurate in terms of quality of the robust solutions obtained
- Disadvantages: a posteriori evaluation of the delay scenarios

Remarks

- Robustness can help reduce issues related to data uncertainty
- Efficiency has to be taken into account
- Two-stage methods directly include scenarios but require significant computational effort
- Single-stage methods are faster but require a posteriori evaluation of the scenarios (e.g., validation tool)